

# Отчет по лабораторной работе №3 — игра «Арканоид»

---

## A. Титульная страница (для каждого участника)

- ФИО студентов: Леонов Богдан Павлович, Зейферт Александр Сергеевич
- Номер студенческого билета: 242063, 242065
- Предмет: Практикум по программированию
- Номер лабораторной работы: 3
- Название игры: «Арканоид»
- Роли в проекте: Lead Developer, Software Engineer

## B. Описание игры

- Реализована игра Арканоид
- Правила: игрок перемещает платформу внизу экрана влево/вправо, чтобы отражать мяч и разрушать все кирпичи; потеря мяча за нижнюю грань уменьшает количество жизней.
- Реализованные изменения и улучшения: случайная прочность кирпичей (1–3 удара) с цветовым кодированием, звуки разрушения/удара, счет и жизни в UI, победа/проигрыш с перезапуском по R, расчет движения по dt для одинаковой скорости на разной частоте кадров.
- Используемые инструменты и технологии: Python 3.10+, Pygame 2.5, dataclasses для конфигурации, JSON для настроек, спрайтовая система Pygame.

## C. Распределение ролей и задач

- Роли участников: Леонов Богдан – Lead Developer; Зейферт Александр – Software Engineer.
- Методы сотрудничества и коммуникации: Git/branches, чат, совместное тестирование.

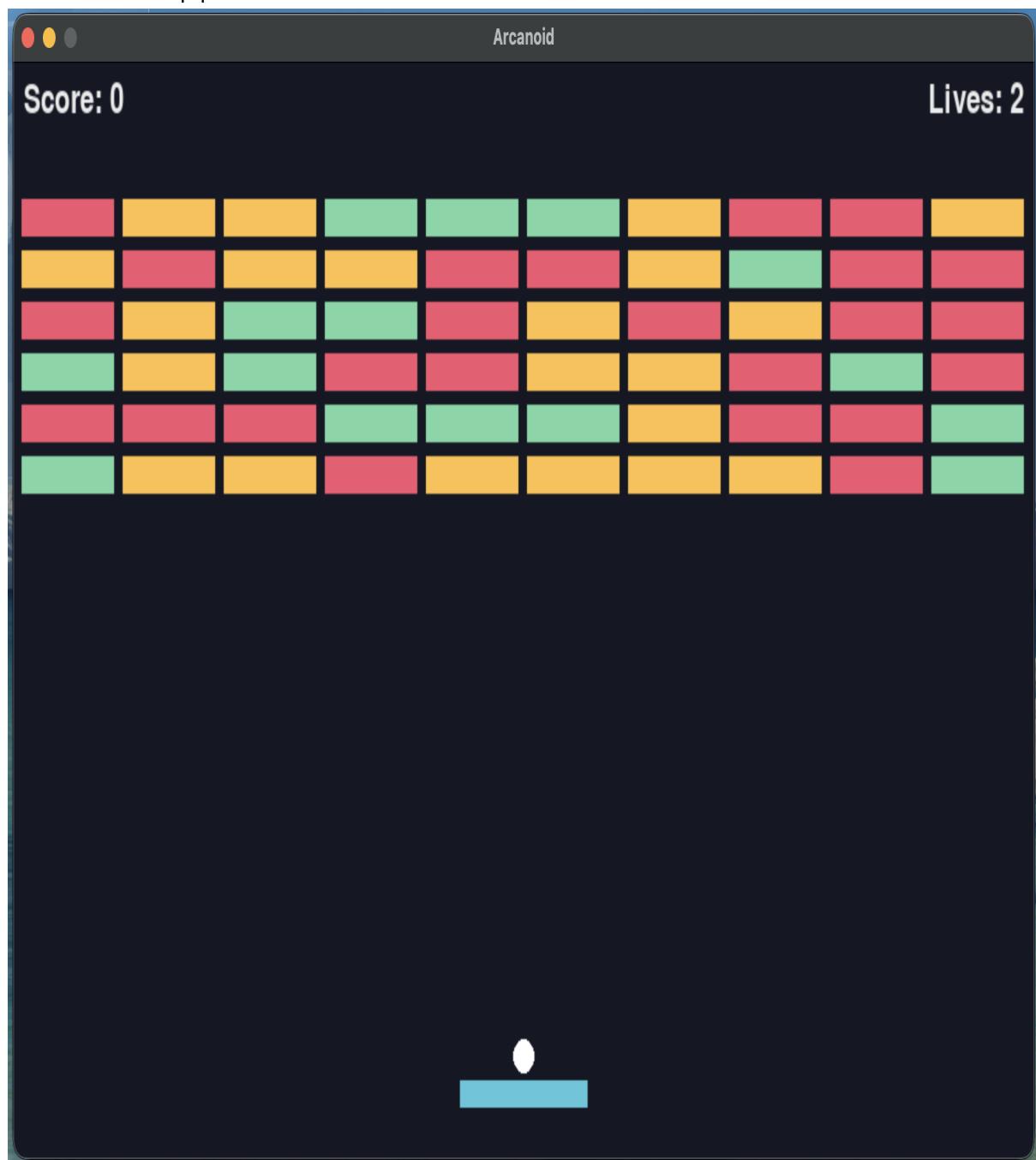
## D. Архитектура проекта

- Структура проекта: arcanoid/ └── main.py └── game.py └── config.py └── config.json └── requirements.txt └── README.md └── report.md └── sprites/ | └── \_\_init\_\_.py | └── paddle.py | └── ball.py | └── brick.py └── sounds/ └── block\_hit.wav └── game\_over.wav └── victory.wav └── гриша.ogg
- Ключевые компоненты и обязанности:
  - Game (game.py): игровой цикл, обработка событий, столкновения, счет, жизни, экраны победы/поражения.
  - Paddle (sprites/paddle.py): движение платформы по вводу, удержание в пределах окна.
  - Ball (sprites/ball.py): старт с платформы, запуск по пробелу, отражения от стен и платформы с вычислением угла, сброс при потере.
  - Brick (sprites/brick.py): прочность, смена цвета, удаление при разрушении.

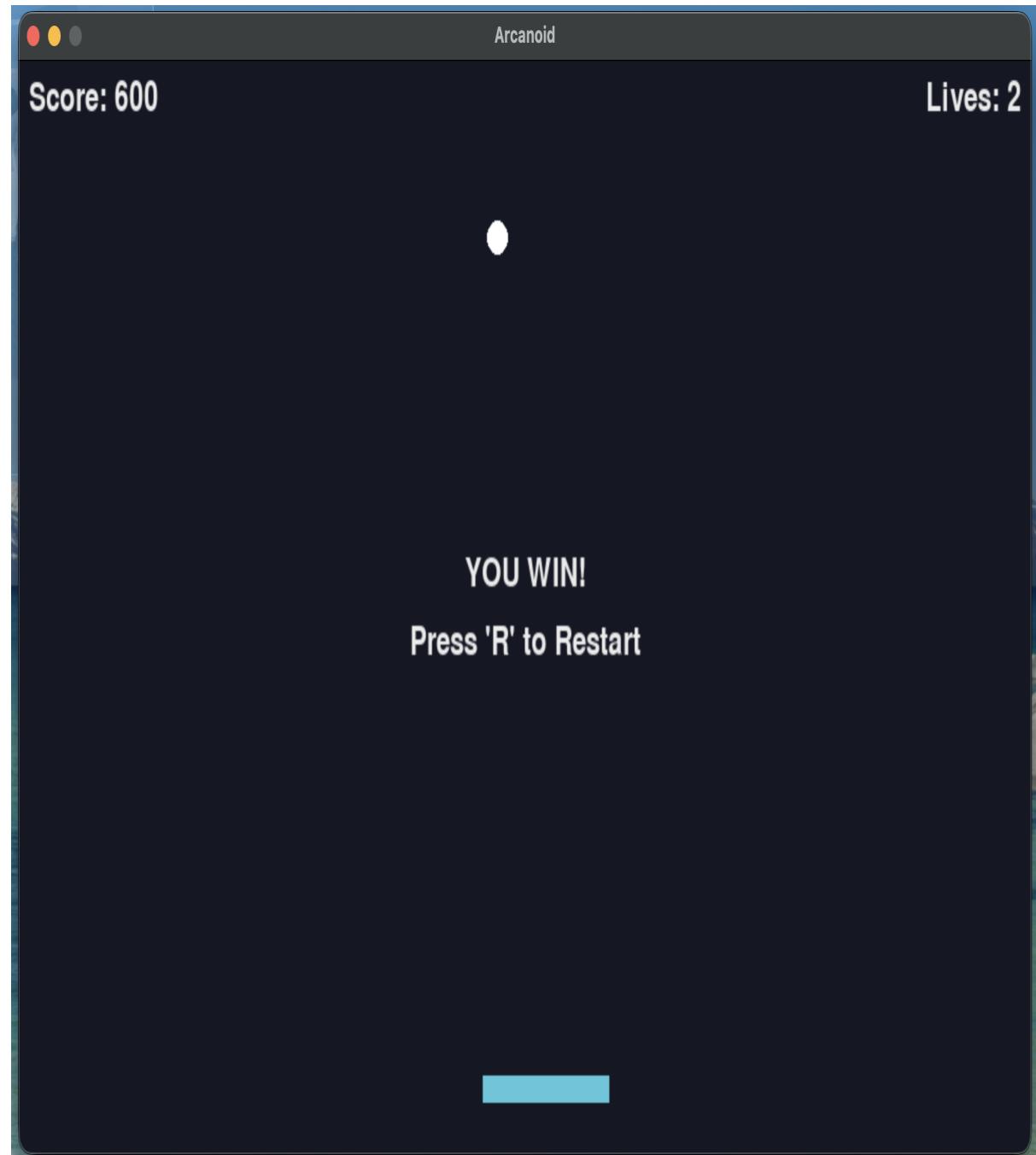
- config.py + config.json: загрузка настроек, палитра, размеры, скорости.
- Использованные шаблоны: конфигурация через dataclass + JSON, разделение ответственности по спрайтам, композиция Game → группы спрайтов, детерминированные апдейты на основе dt.

## E. Реализованный функционал

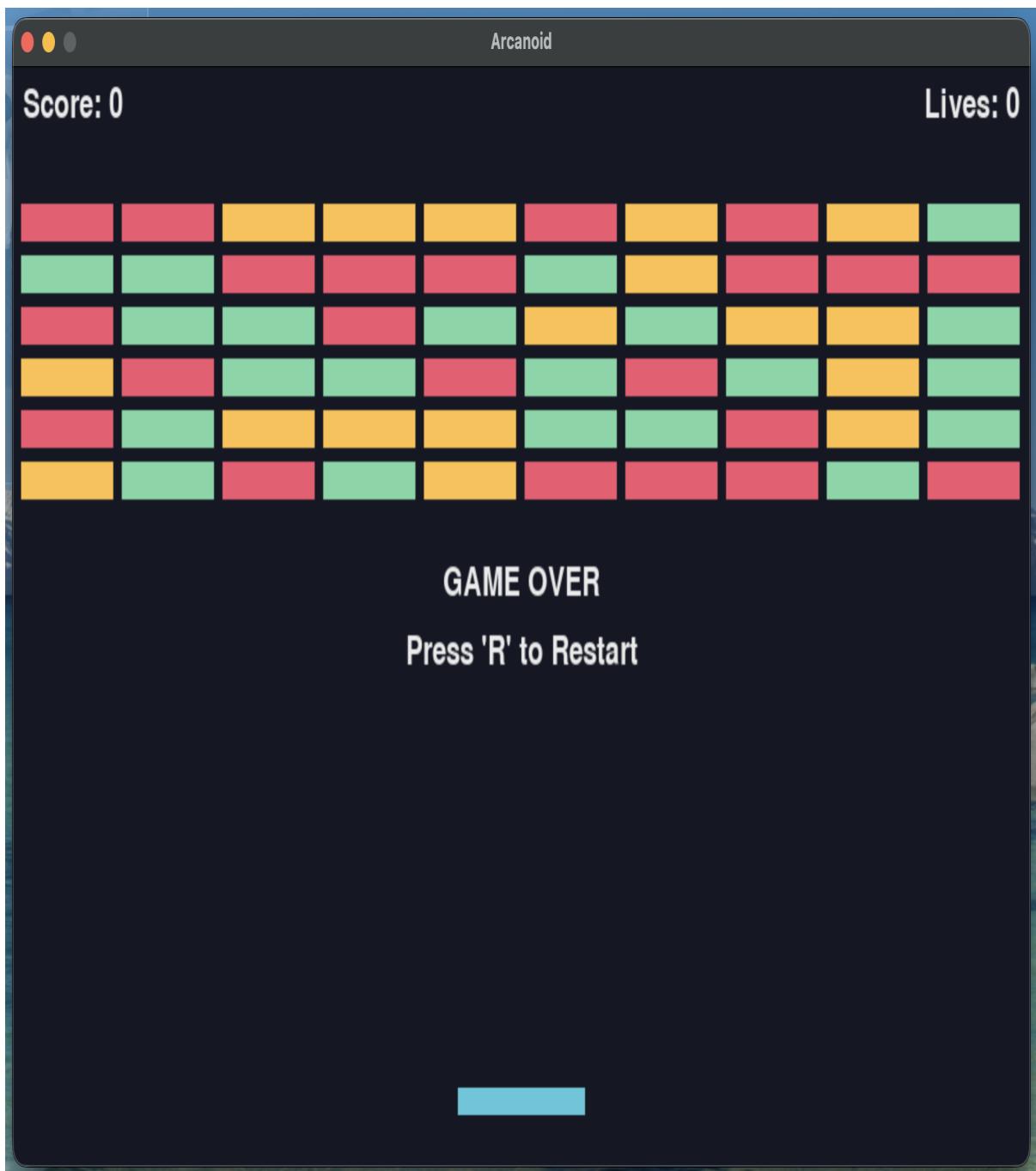
- Основные требования и доказательства:
  - Отрисовка уровня из сетки кирпичей разной прочности (цвета по уровню).
  - Управляемая платформа и мяч со стартом по пробелу, отражение от стен и платформы.
  - Система жизней и счета, победа при очистке поля, проигрыш при исчерпании жизней, перезапуск по R.
  - Звуки для удара, разрушения, победы и поражения.
- Скриншоты:
  - Основной интерфейс:



- Победный экран:



- Экран проигрыша



- Ключевые фрагменты кода:

- Отскок мяча от платформы ([sprites/ball.py](#)):

```
relative_intersect_x = self.rect.centerx - paddle.rect.centerx
normalized_offset = max(-1, min(1, relative_intersect_x /
(paddle.rect.width / 2)))
angle = normalized_offset * math.radians(70)
speed = math.hypot(*self.velocity) or self.speed
self.velocity = [speed * math.sin(angle), -speed *
math.cos(angle)]
```

- Обработка удара по кирпичу и начисление очков ([game.py](#)):

```
for brick in hit_bricks:
    self.ball.velocity[1] *= -1
    destroyed = brick.hit()
    if destroyed:
        self.sound_block_break.play()
        self.score += 10
    else:
        self.sound_block_hit.play()
    break
```

- Сброс мяча при потере жизни (`game.py`):

```
if self.ball.rect.top > C.WINDOW_HEIGHT:
    self.lives -= 1
    if self.lives <= 0:
        self.sound_game_over.play()
        self.game_over = True
    else:
        self.ball.reset()
```

- Решенные технические проблемы: расчет скорости через `dt`, нормализация угла отскока для управляемой траектории, хранение настроек в JSON для быстрой подстройки без изменения кода.

## F. Инструкции по запуску и игре

- Системные требования: Python 3.10+, установленный Pygame; ОС с поддержкой SDL (Windows/macOS/Linux).
- Установка зависимостей: `pip install -r requirements.txt`.
- Запуск: `python main.py` из корня проекта.
- Схема управления: `←/→` — движение платформы; `Space` — запуск мяча; `R` — перезапуск после окончания; `Alt+F4/Cmd+Q` или закрытие окна — выход.
- Правила и цель: разрушить все кирпичи, не потеряв все жизни; каждая потеря мяча уменьшает счетчик жизней, победа наступает при отсутствии кирпичей.

## G. Полный исходный код и ресурсы

- Основные модули:
  - `main.py` — точка входа.
  - `game.py` — игровой цикл, столкновения, UI.
  - `sprites/paddle.py`, `sprites/ball.py`, `sprites/brick.py` — логика спрайтов.
  - `config.py`, `config.json` — настройки размеров/скоростей/палитры.
- Ресурсы: каталог `sounds` (эффекты удара/разрушения/победы/поражения), `sprites/__init__.py` для экспорта классов.
- Структура организации: корень проекта → код (`*.py`), конфиг (`config.json`), ресурсы (`sounds`), зависимости (`requirements.txt`), документация (`report.md`).